浪脚本零起点入门系列(十二)

多边形编辑 Editable Poly

作者：飞浪

本节关键词：Editable_Poly Method, EditablePoly,bitarray

在此系列教程中，本人一直试图把自己学习 maxscript 的经验奉献给读者，当然我的方法肯定不是最好的，但是只要这些经验能帮助更多喜欢 maxscript 的人，本人的目的就达到了。前面的所有章节中，不是所有的地方 100%正确，比如第八节，我建议大家在变量名前面加前缀，后来我在其他书上看到说这个命名方法早就过时了。本人并不是计算机科班出身，对于程序的理解描述有些地并不专业，甚至还会出纰漏，如果你发现有些地方有问题请不要惊讶，还请不吝指教。

maxscript 的语法就像英语的语法，内置函数就像单词，而编写 maxscript 就是把内置函数按照语法写成句子，再按照一定的逻辑排版出来，出来的就是一个完整的代码。这些内置函数全部在 maxscript reference 里面，它就是一部牛津词典。查词典大家都会，不认识的可以用金山词霸，而造句需要有想法。在 maxscript 里面，这些想法并没有程序专业讲的数据结构那么高深，简单的组织一下就可以完成高效的工作。本节涉及到的主要函数在下面两个网页：

Editable_Poly Methods

http://www.cgplusplus.com/online-reference/maxscript-reference/source/editable_poly_methods.htm

Interface: EditablePoly

http://www.cgplusplus.com/online-reference/maxscript-reference/source/interface_editablepoly.htm

下面跟我一起来走近 Editable_Poly 吧！

创建一个茶壶保持选中，运行以下代码：

```
polyOp.getNumVerts $
```

咦？出错啦，得到如下结果：

```
-- Runtime error: EPoly operation on non-Editable Poly: Teapot
```

它说 EPoly 的操作运行在非 Editable Poly 的物体上导致出错，所以本节所讲的操作全部要在 Editable Poly 下运行啊。

那我们就把它转成 EPoly：

```
convertTo $ Editable_Poly
```

再运行第一个代码 polyOp.getNumVerts $

得到结果 530，这个是此茶壶的点的总数量。看上面那句代码，polyOp 是一个已经编写好的结构，里面包括了许许多多的 EPoly 操作函数，运行一下 polyOp 得到如下结果：

```
#Struct:polyop(
  getEDataChannelSupport:<fn>,
  getMapFace:<fn>,
  makeEdgesPlanar:<fn>,
  capHolesByFace:<fn>,
  makeVertsPlanar:<fn>,
  setSlicePlane:<fn>,
  getEdgeVerts:<fn>,
  getDeadVerts:<fn>,
  getNumVerts:<fn>,
  ......
```

调用这些函数只需要用点语法即可。函数的说明在本文开头介绍的网页里面。

polyOp.getNumEdges $是获取边的总数，polyOp.getNumFaces $是获取面的总数。下面我用场景助手里面的随机选择点、线、面、元素来说说 EPoly 的基本操作。

选择茶壶的一些点，运行代码：

```
polyOp.getVertSelection $
```

得到类似于下面的结果：

```
#{61..65, 129..132, 193..196, 246..247, 517..518}
```

这些是什么东东？getVertSelection 的作用是获取选中的点，返回的就是这些点的信息了，那这结果有什么意义呢？其实我们得到的结果是一个 bitarray，它是另一种数据常量，前面没有细讲。这里顺带讲一下。bitarray 字面意思是位数组，我们知道数组#(1，2，3)用的是中括号，而位数组用的是大括号：#{1,2,3}；数组里面的元素可以是任何数据，而位数组里面只能是正数。bitarray 里面的数字只是一个开关，

表示在此 bitarray 里面，此位置是 true 还是 false。怎么理解呢？bitarray 对子元素的访问跟 array 一样，如有以下代码：

```
bitarr=#{1,3,5,6,7}

for i in 1 to 7 do print bitarr[i]
```

得到的结果是 true,false,true,false...，上面的#{1,3,5,6,7}表示 1,3,5,6,7 这四个位置是开启的，所以为 true，用在 EPoly 可以表示选中了第 1,3,5,6,7 个点。而其他的数字在上面的 bitarray 里面都是 false 在 EPoly 里面可以表示这些点没有选中。可以用两个小点表示两个及以上的连续数字，如上面的 bitarr 可以写成#{1,3,5..7}。现在再回头看看上面得到的点的信息：#{61..65, 129..132, 193..196, 246..247, 517..518}，这个 bitarray

在 61 到 65 的位置是 true，所以这些点是选中的，后面类推。下面是对 bitarray 一个直观的解释：

```
bitarr=#{1,3,5..7} --假设有一排灯，分别以 1，2，3，...编号，这个数组就表示只打开了 1,3,5,6,7 号灯
bitarr.count --表示此 bitarr 中的最大的数字（打开的灯中最大的编号），在这里是 7，注意这里跟 array 不同
bitarr.numberset --表示此 bitarr 中所有为 true 的元素的数量（所有打开的灯的数量），这里才是 array 里面的 count
bitarr[1]=false --把 1 号位的灯关掉
bitarr[12]=true --把 12 号位的灯打开
bitarr.count --此时打开的灯中最大的编号变成了 12
bitarr as array --把 bitarr 转变成了 array，自己观察一下结果
#(1,2,3,55,57,98,99,100) as bitarray --当然 array 也可以转变成 bitarray，但是 array 里面的元素必须为正数
```

回到 EPoly 中来，怎么样选中自己设定的点呢？执行如下代码：

```
polyOp.setVertSelection $ #{1..100} --选中 EPoly 的第 1 到 100 个点，切换到点层级就能看到效果
```

现在我们知道了点的总数量用 polyOp.getNumVerts，选择点用 polyOp.setVertSelection，那什么，来个随机选择点呗，来啦：

```
vertTotal=polyOp.getNumVerts $ --点总数
percent=0.3 --选择 30%
selVert=#{} --要选择的点的 ID，这里是空的
while selVert.numberset < vertTotal*percent do
( --当点数小于总数的 30%时，不断的随机出数字，将 selVert 中对应的点设置为 true
selVert[random 1 vertTotal]=true
)
polyOp.setVertSelection $ selVert --选择点
subobjectlevel = 1 --设置当前选择物体的子层级为 1，EPoly 即为点层级
update $ --刷新一下物体，马上看到效果
```

边的总量用 polyOp.getNumEdges，选择边用 polyOp.setEdgeSelection，子层级为 2，把上面的代码替换一下就可以随机选择边了，同理面是 polyOp.getNumFaces，选择用 polyOp.setFaceSelection ，子层级为 4，替换代码即可选择面，这些函数都从 reference 中来，没有什么高深的。下面来看看 EPoly 中的 element。打开 macro recoder，在 element

层级选择壶嘴，可以看到如下代码：

$.EditablePoly.SetSelection #Face #{321..384}

这是选择了 ID 号从 312 到 384 的面，如果用 polyOp 表示就是：

```
polyOp.setFaceSelection $ #{321..384}
```

壶嘴是一个元素 element，但是显示的为什么还是选择面呢？其实，element 就是一些 ID 号连续的面的集合，element 本身是没有 ID 的，我们有办法用脚本选择指定的 element 吗？答案当然是有啦。在本文的开头介绍的第二个网页里面，有一个超级用的的函数 SelectElement()，它可以根据当然选择的面，选择与这些面相关的 element。执行如下代码：

```
polyOp.setFaceSelection $ #{1} --选择 ID 号为 1 的面
$.EditablePoly.SelectElement() --选择与选择的面为同一 element 的面，是不是有点绕口？
```

这样就选择了包括 ID 号为 1 的 element。那其他的 element 呢？既然 element 自身没有 ID，那么我们就用 face 的 ID 号来标记，每个 element 只需要取出一个 face ID 出来就行了。理解一下下面的函数：

```
fn getIDsOfElement obj =
( --此函数用来获取每个 element 中的第一个 face ID
select obj
faceNum=polyOp.getNumFaces $
ids=#{1} --储存 face id
tempID=1
while tempID <= faceNum do
( polyOp.setFaceSelection $ #{tempID}
  $.EditablePoly.SelectElement()
  selFace=polyOp.getFaceSelection $
  ids[tempID]=true
  tempID+=selFace.numberset
)
ids as array --把结果转换成数组，便于使用
)
```

有了上面这个函数，我们就可以对 element 像对待 face 一样的操作了，对茶壶执行此函数：

```
getIDsOfElement $
```

得到结果#(1, 257, 321, 385)

此结果里面的数字是 face ID，每个 ID 代表着一个 element，那么对 element 随机操作也变得简单了：

```
eleID=getIDsOfElement $ --获取与 element 相关的 face ID

percent=0.5 --选择 50%的 element

selID=#() --需要选择的 face ID

while selID.count < eleID.count*percent do

(

tempID=eleID[random 1 eleID.count] --随机抽取一个 ID

if finditem selID tempID == 0 then append selID tempID

)

polyOp.setFaceSelection $ (selID as bitarray) --选择这些面，记得要转换成 bitarray

$.EditablePoly.SelectElement()
```

上面讲了点、线、面、元素，EPoly 的第三个层级没说，就是 Border，它表示的是只有一个面用过的边，孤独的存在着，我们也不能把它遗忘啊，下面的代码直接选择这些边：

```
subobjectlevel=3

myEdges=polyOp.getOpenEdges $

polyOp.setEdgeSelection $ myEdges
```

到这里，相信大家对 EPoly 在 maxscript 中的操作已经有了一个认识了吧，本节所用到的函数只是 EPoly 操作中的几个而已，开头提到的两个网页的内容也不是全部的 EPoly 操作，帮助里面 Frequently Asked Questions 也有关于 EPoly 的例子，还是那句，函数都在帮助文档里面，你需要做的就是找出自己需要的函数然后按一定逻辑编写在一起，CG++里面的高手在逐渐增多，希望大家自由交流，快乐编程：）

补充：下面代码是分离出所有的 Epoly，在老外的代码基础上更改过来的，polyop.getElementsUsingFace 函数以前没发现。原贴地址：http://forums.cgsociety.org/showthread.php?f=98&t=785636 原来代码在 15 楼。更改后的代码：

```
fn separateByFaceElements obj:selection[1] = if iskindof obj Editable_Poly do

(

local elements = #()

local faces = obj.faces as bitarray

while (f = (faces as array)[1]) != undefined do
```

```
(
ff = polyop.getElementsUsingFace obj #{f}
append elements ff
faces -= ff
)
local nodes = #()
for k=1 to elements.count do
(
maxops.CloneNodes obj newNodes:&nn
newname=uniqueName obj.name
polyOp.detachFaces nn[1] elements[k] delete:on asNode:on name:newname
centerPivot (execute ("$'"+newname+"'"))
-- update nn[1]
delete nn[1]
)
delete selection[1]
)
```

选择物体后运行：separateByFaceElements()

本节原帖地址：http://www.cgplusplus.com/bbs/viewthread.php?tid=2310

本节结束。